Autonomic Computing Correlation for Fault Management System Evolution

Autonomic Computing Correlation for Fault Management System Evolution

Roy Sterritt[1], Dave Bustard[2], Andrew McCrea
[1]School of Computing and Mathematics, [2]School of Computing and Information Engineering,
Faculty of Engineering, University of Ulster,
Northern Ireland.
E-mail: [1]R.Sterritt@ulster.ac.uk, [2]DW.Bustard@ulster.ac.uk

Abstract

This paper discusses the emerging area of autonomic computing and its implications for the evolution of fault-management systems. Particular emphasis is placed on the concept of event correlation and its role in system self-management. A new correlation analysis tool to assist with the development, management and maintenance of correlation rules and beliefs is described.

1. Introduction

Autonomic computing [1] is rapidly becoming established as a significant strategic approach to the design of computer based systems. Its envisaged goal is the production of systems that are self-managing in four main respects: self-configuring, self-healing, self-protecting and self-optimising.

Self-managing systems should be more robust and autonomous, reducing their total cost of ownership. In the short-to-medium term, however, the modification of existing systems to include autonomic functionality is likely to increase maintenance costs, offset by suitable tools and processes to assist with this task.

When launching autonomic computing as a new strategic direction, IBM highlighted the growing complexity crisis in the IT industry, comparing it with telephony in the 1920s. There, the rapid increase in use of the telephone led to estimates that by the 1980s half of the population of the USA would have to be employed as telephone operators to meet the demand [1]. The implementation of automated switching and other technological developments avoided this crisis. By analogy, IBM is expecting autonomic system implementations to achieve similar productivity gains. It is anticipated, however, that significant research and development will be required to achieve that goal.

This paper considers the self-healing aspect of autonomic computing and even more specifically focuses on the analysis of fault events in distributed systems. It describes a correlation prototype tool to assist with the discovery of new rules, correlations and beliefs in fault alarms. The particular domain used to motivate the discussion is fault management systems in telecommunication networks. Although such commercial networks achieve high reliability (99.999%) [2], their growing complexity can benefit from an autonomic approach.

2. Overview of Telecom Survivable Networks

Since the 1920's, automation in telephony has evolved substantially. The Internet, with its vast infrastructure supporting millions of interconnected computers is perhaps the most significant development. The complexity of networks has grown in various ways [3]. As user demands and expectations become more varied and complex so do the networks themselves. Data, voice, image, and other information now travels under the control of different protocols through numerous physical devices manufactured and operated by different vendors. It is expected that the trend towards increasing complexity will continue.

Several factors contribute to this situation such as the increasing complexity of individual network elements, the need for sophisticated services and the heterogeneity of connected equipment [5].

The systems are designed to be robust since it is simply not acceptable for millions of calls to be cut-off due to a faulty network element or a software upgrade. This leads to design approaches that incorporate back-up mechanisms that allow for recovery from certain classes of fault. One technique, for example, is the use of a ring topology for node connection as illustrated in Figure 1. In SDH/Sonet systems, traffic travels in both directions. Any fault occurring that prevents progress in one direction will cause an automatic switch in traffic direction to avoid the failure area, thus sustaining traffic throughput.

This fits with the autonomic goal that there should be no failure at the system level. Components of the system will fail but self-configuration is used to ensure minimal disruption [22].

0-7803-8200-5/03/$17.00 ©2003 IEEE

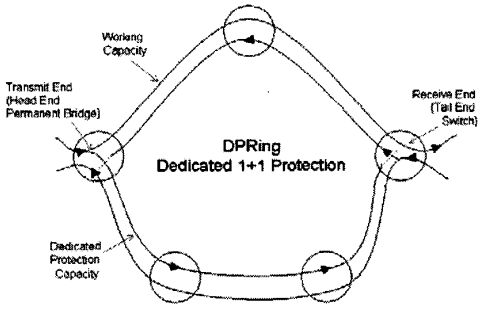


Figure 1 Survivable Network Architectures

For major hub traffic applications, survivability tends to be implemented through an additional dedicated protection ring (Figure 1). In metropolitan, junction and trunk network applications this robustness may be achieved through the less expensive option of a shared protection ring, which reserves protection capacity in the existing ring in case of failure.

Robustness, in general, is achieved through redundancy in the hardware and software components of the network. Unfortunately this can increase complexity even further, made worse by allowing (old) non-synchronous traffic to co-exist with synchronous traffic.

Central to the management of these complex networks is processing of event messages. By analogy with the human autonomic nervous system these are similar to the electric pulses that travel along nerves. When a fault occurs in an SDH network a series of triggered events are usually reported to the element controller (manager). The behavior of the alarms is often so complex it appears non-deterministic [6], making it very difficult to isolate the true cause of the fault [7]. Failures in the network are unavoidable but quick detection and identification of their source is essential to ensure robustness. The correlation of alarm event messages is an important part of this analysis [8]. The major telecommunication equipment manufacturers deal with event correlation through alarm monitoring, filtering and masking as specified by ITU-T [9] and other international standard bodies. Resulting rule type diagnostic systems provide assistance to the operator whose expertise is then used to determine the underlying fault (or faults) from the filtered set of alarms reported.

Currently, the skill of the operator is central to identifying faults. So although automation prevents the immediate loss of traffic and preserves the general function of the system, intervention is necessary to determine and resolve problems that arise. The promise of autonomic computing is a significant reduction in the role of the operator.

3 Autonomic Computing System Architecture

The basic building blocks of any autonomic system architecture must include *sensors* and *effectors* [10]. By monitoring behavior through sensors, comparing this with expectations (historical and current data, rules and beliefs), planning what action is necessary (if any) and then executing that action through effectors, creates a control loop [11]. The control loop, a success of manufacturing science for many years, provides the basic backbone structure for each system component [22].

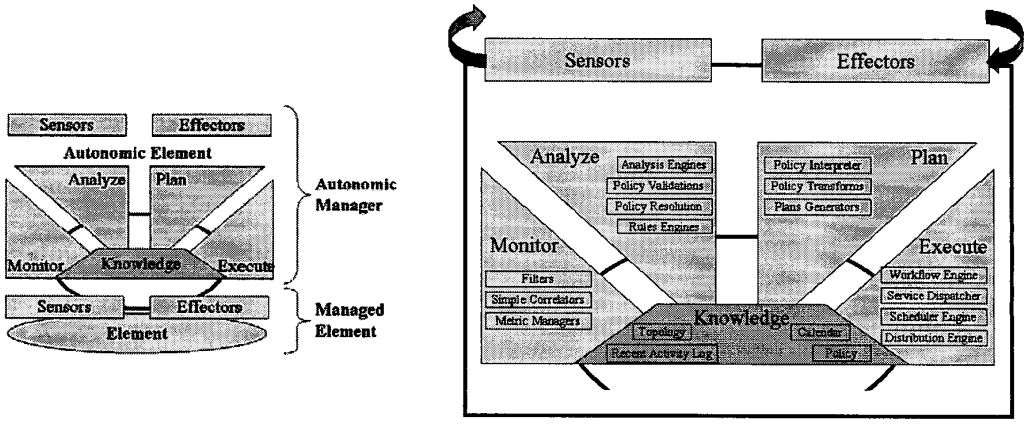
Figure 2 is IBM's view of the necessary components within an autonomic manager. (For an alternative artifacts view, see [23].) It is assumed that an autonomic manager is responsible for a managed element within a self-contained autonomic element. Interaction will occur with remote autonomic managers through virtual, peer-to-peer, client-server [12] or grid [13] configurations (see Figure 3).

The *monitor* and *analyze* parts of the structure process information from the sensors to provide both self-awareness and an awareness of the external environment. The *plan* and *execute* parts decide on the necessary self-management behavior that will be executed through the effectors.

The *simple correlator* in the *monitor* parts and the *rules engine* in the *analyze* part use correlations, rules, beliefs, expectations, histories and other information known to the autonomic element, or available to it.

There are two strategies for introducing autonomic behaviour. The first is to engineer it into systems and the second is to achieve it through adaptive learning. The first approach can be taken now, with human experts generating or overseeing the generation of rules for autonomic functions. Over time, this could be increasingly supplemented with self-learning processes [14].

Work is currently underway to add autonomic capabilities to legacy systems, in areas such as instant messaging, spam detection, load balancing and middleware [15].



(a) General concept of an Autonomic Element

(b) Necessary Components within the Autonomic Manager

Figure 2 IBM's view of the Architecture of an Autonomic Element

4. Correlation

The introduction of autonomic principles requires the monitoring of individual system components through sensors and the ability of those components to respond to requests through effectors. Monitoring will typically involve the *correlation* of several related pieces of information. Correlation is important in both self-assessment and in the assessment of a component's operating environment. This helps in deciding when action is required and what should be done.

Figure 3 depicts a logical autonomic environment where each self-contained autonomic element, consisting of the autonomic manager and the managed component, monitor an autonomic signal channel to receive information about the changing environment and to report changes that may affect the environment.

Event correlation is a conceptual interpretation of multiple events, giving them a collective meaning. This produces a new higher-order compound event that helps determine what action is required. Jakobson and Weissman describe correlation as a generic process involving six operations: *compression*, *suppression*, *count*, *Boolean patterns*, *generalization*, and *specialization* [8]. These are defined as follows:

Compression [A,A,....A] => A
Multiple occurrences of an event (A) can be compressed into a single event.

Suppression: [A, B, p(A)<p(B)] => Ø
A low-priority event (A) may be inhibited in the presence of a higher-level event (B).

Count [n x A] => B
A specified number (n) of occurrences of an event can be substituted with a new event.

Boolean Pattern [A, B, ... T, ∧,∨,¬] => C
A new event can be substituted for a set of events satisfying a Boolean pattern.

Generalization [A, A ⊂ B] => B
An event (A) can be generalized to its super class (B).

Specialization [A, A ⊃ B] => B
An event (A) can be specialized to a sub-class (B).

The next sections look briefly at rule discovery, followed by the consideration of a new correlation tool with specific application within telecommunications fault management systems.

5. Rule Discovery

The principle aim of event correlation is the interpretation of the events involved. The event signals or messages represent *symptoms*. Rules and beliefs identify which events to correlate and how they should be transformed. These tend to vary over time creating a significant maintenance burden [16]. Machine learning, data mining and other AI techniques can assist in the discovery of correlation rules and beliefs [19][20]. However, a human-centred process is more effective than either a human or computer operating independently [18].

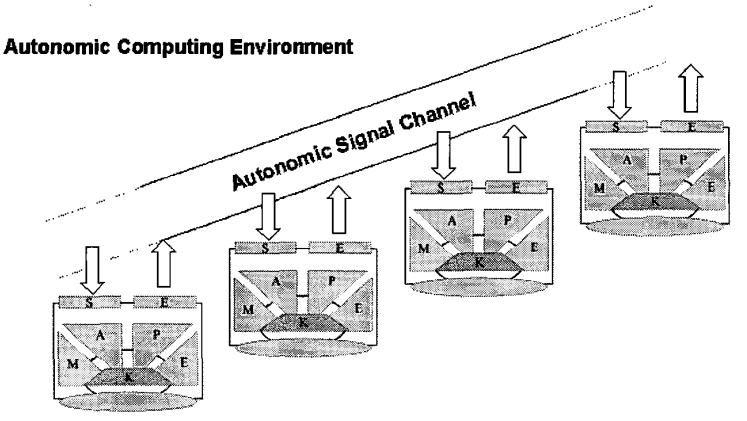


Figure 3 Autonomic Computing Environment

In previous work, correlation rule discovery was described using of a three-tier architecture model [21]:

Tier 1 – Visualization of Event Messages
The visualization tier allows visualization of the data in several forms. It provides data interpretation and evaluation throughout the knowledge discovery process, from data cleaning to data mining.

Tier 2 - Managing the Correlation Rules
The second tier supports the definition of correlation rules that are discovered by experienced operators.

Tier 3 – Discovering Correlation Rules
The third tier mines the telecommunications management network messages to produce more complex correlation rules.

This three-tier architecture enables both computer-aided human discovery and human-aided computer discovery and shows how an integrated solution consisting of such different components as a visualization tool, rule-tool and machine-learning tool can produce a very useful fault-management solution.

The correlation analysis tool, discussed in the next section, fits within tier 2 while also providing some visual feedback. It is capable of testing and executing discovered rules on event data, a vital task in testing non-trivial rules.

6. AC Correlator Analysis Tool (acCAT)

A survivable network architecture attempts to ensure continued service but does not necessarily determine the fault without human intervention. Self-diagnosis is obviously a prerequisite for self-healing. The standard correlation approach is through a simple three-stage process of monitoring, filtering and masking of the alarm events.

In practice, there may be a large number of uncorrelated alarm event messages on a network at any one time. One estimate of BT's UK network, for example, is that on average 95% of all alarm events raised remain uncorrelated. At any moment in time this represents tens of thousands alarm events. These are collected, providing a body of material for subsequent data mining. This is used to reveal correlation rules or identify patterns that can help further automate the fault identification process to reduce the number of uncorrelated events.

The acCAT prototype is an interactive tool to test and execute discovered correlation rules using the six transformation rules identified in the previous section: *compression*, *suppression*, *count*, *Boolean patterns*, *generalization*, and *specialization*.

Figure 4 shows the high-level structure of the tool. The inference engine encompasses:

1. the *user interface*—through which the user is able to influence the analysis strategy;
2. the *control process*—which controls the sequencing of the strategy and the components which carry it out; and
3. the *correlation engine*—which contains the lower level components for performing the correlation.

The knowledge base encompasses the rule base processing, which is responsible for maintaining rules, and, to a lesser degree, the user interface where changes to rules can be made.

The rule base contains the correlation rules that can be applied to the system. This is kept and maintained as

a protected system file. The original rules are recoverable through a backup rule base. To facilitate the addition of new rules discovered from other components within the three-tier architecture, XML is used as the general rule format.

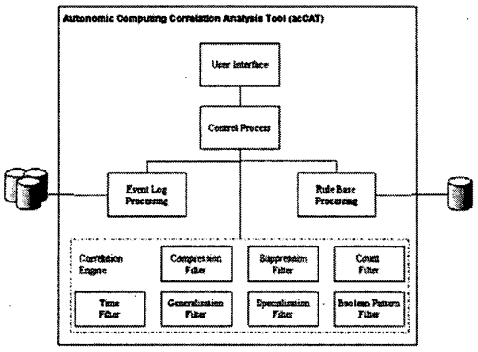


Figure 4 High-Level design architecture for acCAT

The user interface is responsible for managing all interactions with the user. It uses the API provided by the control process to perform all operations and is not directly aware of any of the underlying classes. Screenshot 1 to Screenshot 3 demonstrate some of the functions of the tool. The control process provides methods to access alarms and objects. File processing is conducted from here, and it contains EventList, RuleList and CorrelationEngine objects which control the flow of data among these processes and also between the objects and the user interface. These objects contain the 'knowledge' of the system.

The Log Processing object is responsible for taking in data from the Event Logs and creating Event Objects.

Rule Base processing, like Log File Processing, is responsible for reading from a file and creating objects—in this case, Rule Objects. As rules can be created, edited and deleted this component requires full privileges to the Rule Base database. Rule Base processing is also responsible for allowing access to the Rule Objects.

The correlation engine contains all the filters required to perform the correlation. The EventList and RuleList (where required) are passed between these filters resulting in the return of a correlated EventList. The engine contains seven filters, each partially configurable. These filters include a Time Filter and filters for the six generic correlation transformations described above.

Essentially acCAT can take discovered rules from tier 1 – visualization or tier 3 – mined rules (if in XML format) and allow a user to experiment by applying the rules to event logs to see the effects of the new correlations (note the colored visual indicators on the right-hand side of the screenshots).

acCAT can also be used directly by an expert who may have implicit or tacit knowledge about how the system works to develop that knowledge into rules and experiment on the alarms to see the resultant effects.

The tool may also assist in the purpose of debugging as well as managing discovered rules and testing these and existing rules against new network equipment and situations.

The developed rules (be they of compression, suppression, count, Boolean patterns, generalization, or specialization type) may then be incorporated into the telecom management system to facilitate fuller automation on the road to achieving an autonomic system.

7. Related Work

Several tools with similar global aims to the three-tier rule discovery architecture and acCAT, have recently been released by IBM through their AlphaWorks autonomic zone website [24].

A tool with similar objectives to acACTS is the generic Log and Trace Tool that correlates event logs from legacy systems to identify patterns. These can be used to facilitate automation or help in debugging.

The Tivoli Autonomic Monitoring Engine essentially provides server level correlation of multiple IT systems to assist with root cause analysis and automated corrective action.

The ABLE rules engine can be used for more complex analysis. In effect it is an agent building learning environment that includes time series analysis and Bayes classification among others. It correlates events and invokes the necessary action policy.

These tools can then be complemented with a policy tool for policy-based management that sets out to reduce the complexity of product and system management by providing uniform cross-product policy definition and management infrastructure [22].

8. Conclusion

The Autonomic computing initiative is starting to gain ground as an approach to computer system development. It brings together many existing research disciplines, with the aim to create robust systems based on a model of self-managing biological systems.

The majority of the initial interest has been on self-optimisation as this may produce the best immediate

return for effort [22]. For autonomic computing to achieve its aim, the other aspects of the approach must be addressed adequately.

In this paper a high-level autonomic computing environment was discussed and the need for monitoring and correlation of events from the internal and external environment was highlighted.

The telecommunications industry was studied as a system that has extensive robustness. The telecommunications approach to fault handling attempts to ensure under reasonable circumstances that the functionality of the system continues. Yet the approach does not necessarily identify the actual underlying fault preventing the self-healing and self-managing goals of Autonomic Computing being fully realised.

A correlation analysis prototype tool was presented that assists with semi-automated discovery and maintenance of rule base that will be key for achieving autonomic behaviour.

In essence such a tool should only have a short-medium term life span to assist in engineering autonomic functions into systems. Effectively as the system evolves in autonomicity, the system itself should increasingly take control of its own rules, beliefs and policies refining these through self-optimising and self-configuration making the tool redundant.

In such a scheme, the visualisation aspects in the three-tier architecture should be adapted to provide human insight into how the autonomic system is functioning ensuring understanding and trust.

Acknowledgements

This work was undertaken through the Centre for Software Process Technologies, which is supported by the EU Programme for Peace and Reconciliation in Northern Ireland and the Border Region of Ireland (PEACE II).

The telecommunications research was undertaken in the Jigsaw project (ITS Start 187), funded by Nortel Networks' Belfast Labs and the Industrial Research and Technology Unit (IRTU).

This research is currently being further explored under a BT Exact Short Term Research Fellowship (2003).

References

[1] P. Horn, "Autonomic computing: IBM perspective on the state of information technology", IBM T.J. Watson Labs, NY, 15th October 2001. Presented at AGENDA 2001, Scotsdale, AR. (available http://www.research.ibm.com/autonomic/), 2001

[2] J. Gray, talk on "Dependability in the Internet Era" at High Dependability Computing Consortium. May 2001

[3] T. Oates. Fault identification in computer networks: A review and a new approach. Technical Report 95-113, University of Massachusetts at Amherst, Computer Science Department, 1995.

[4] C. Bournellis. Internet '95. Internet World, 6(11):47--52, 1995.

[5] M. Cheikhrouhou, P. Conti, J. Labetoulle, K. Marcus, Intelligent Agents for Network Management: Fault Detection Experiment. In Sixth IFIP/IEEE International Symposium on Integrated Network Management, Boston, USA, May 1999.

[6] A. T. Bouloutas, S. Calo, and A. Finkel, "Alarm correlation and fault identification in communication networks", IEEE Trans. on Comms., vol. 42, no. 2/3/4, 1994.

[7] M. Klemettinen, "A knowledge discovery methodology for telecommunication network alarm databases", Ph.D. Thesis, University of Helsinki, Finland, 1999.

[8] G. Jackobson and M.D. Weissman, "Alarm correlation", IEEE Network, vol. 7, no. 6, pp. 52-59, Nov. 1993.

[9] ITU-T Recommendations, M.3010 principles for a telecommunications management network, Feb. 2000.

[10] A.G. Ganek and T. A. Corbi, The dawning of the autonomic computing era, IBM Systems Journal, Vol. 42, no. 1, 2003, pp 5-18

[11] Autonomic Computing Concepts, IBM White Paper

[12] D.F. Bantz, C. Bisdikian, D. Challener, J.P. Karidis, S. Mastrianni, A. Mohindra, D. G. Shea, and M. Vanover, Autonomic personal computing, IBM Systems Journal, Vol. 42, no. 1, 2003, pp 165-176

[13] G. Deen, T. Lehman, J. Kaufman, "The Almaden OptimalGrid Project", Proceedings of the Autonomic Computing Workshop, 5th Int. Workshop on Active Middleware Services (AMS 2003), Seattle, WA, pp 14-21, June 2003.

[14] R. Sterritt, (Dec 2002) "Towards Autonomic Computing: Effective Event Management", Proceedings of 27th Annual IEEE/NASA Software Engineering Workshop (SEW), Maryland, USA, December 3-5 2002, pp 40-47

[15] G. Kaiser, J. Parekh, P. Gross, G. Valetto, "Kinesthetics extreme: An External Infrastructure for Monitoring Distributed Legacy Systems", Proceedings of the Autonomic Computing Workshop, 5th Int. Workshop on Active Middleware Services (AMS 2003), Seattle, WA, pp 22-30, June 2003.

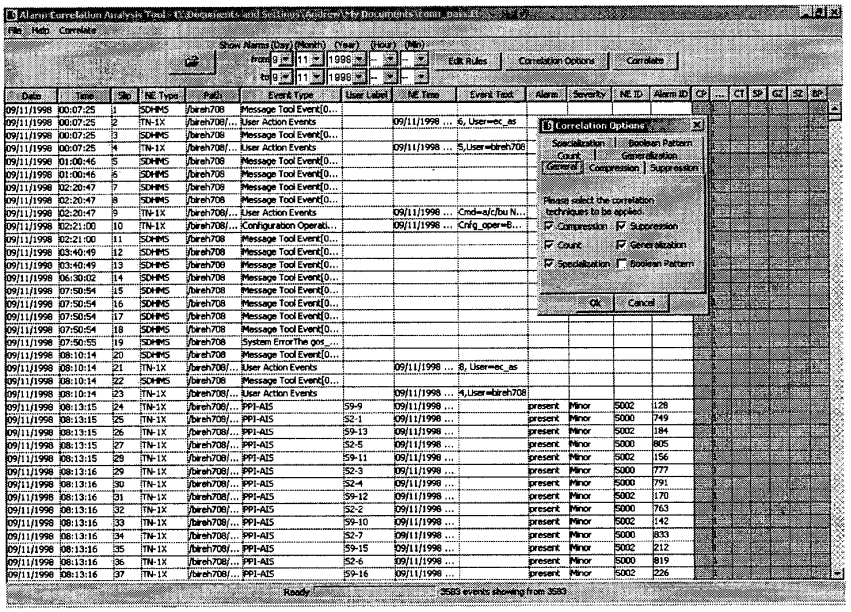
[16] Bratko, S. Muggleton, "Applications of Inductive Logic Programming", Communications of the ACM, Vol. 38, no. 11, 1995, pp 65-70.

[17] R.J. Brachman, T. Anand, "The Process of Knowledge Discovery in Databases: A Human-Centered Approach", Advances in Knowledge Discovery & Data Mining, AAAI Press & The MIT Press: California, 1996, pp 37-57.

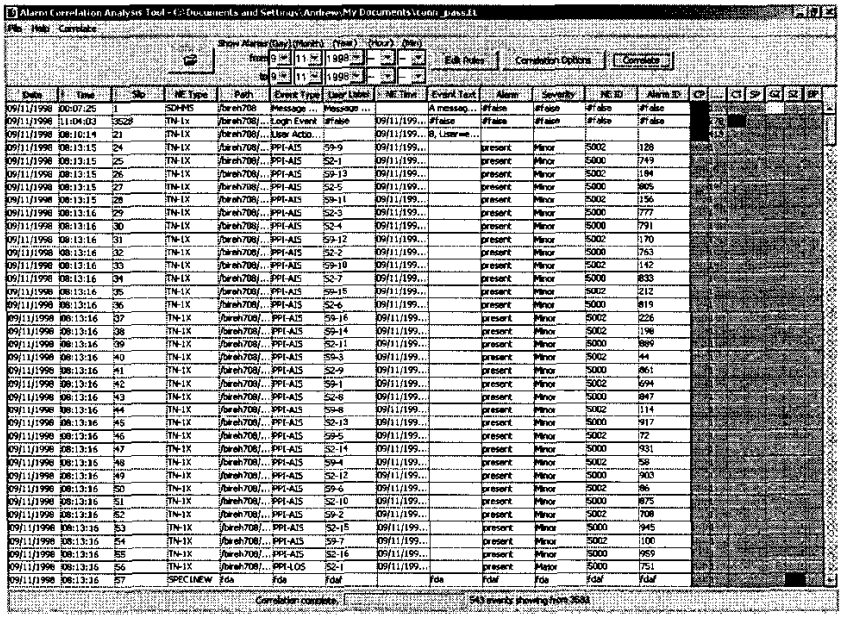
[18] R. Uthurusamy, "From Data Mining to Knowledge Discovery: Current Challenges and Future Directions", Advances in Knowledge Discovery & Data Mining, AAAI Press & The MIT Press: California, 1996, pp 561-569.

[19] R. Sterritt, D.W.Bustard, Fusing Hard and Soft Computing for Fault Management in Telecommunications Systems, IEEE Trans. Systems Man and Cybernetics part C, 32(2)

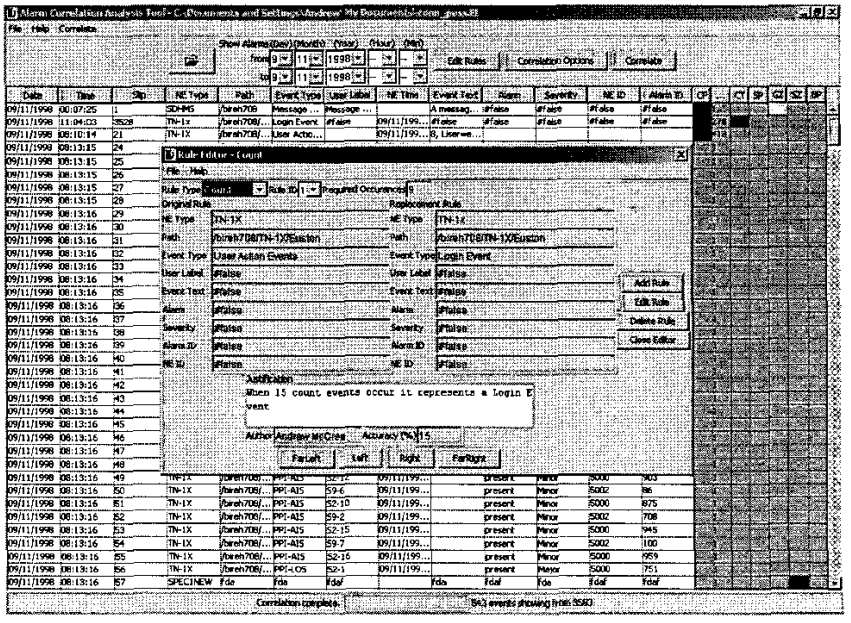
[20] R. Sterritt, "Facing fault management as it is, aiming for what you would like it to be", Soft-Ware: 1st International Conference on Computing in an Imperfect World, Belfast 8-10 Apr., LNCS2311, Eds. D.W. Bustard, W. Liu, R. Sterritt, Springer-Verlag, 2002
[21] R. Sterritt, (Apr 2001) "Discovering Rules for Fault Management". Proceedings of Eighth Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS '01)
[22] A. Ganek, "Autonomic Computing: Implementing the Vision", Keynote presentation at the Autonomic Computing Workshop, 5th Int. Workshop on Active Middleware Services (AMS 2003), Seattle, WA, 25th June 2003.
[23] R. Sterritt, D.W. Bustard, "Towards an Autonomic Computing Environment", 1st International Workshop on Autonomic Computing Systems at 14th International Conference on Database and Expert Systems Applications (DEXA'2003). Prague, Czech Republic Sept. 1-5, 2003.
[24] IBM, alphaworks Autonomic Computing site, http://www.alphaworks.ibm.com/autonomic



Screenshot 1 Viewing an Event Log and the Correlation Options



Screenshot 2 Correlation Results showing events that have been correlated



Screenshot 3 Correlation Rule Editor